Method for the coding/decoding of VLIW cached instructions

The invention relates to a method for actuating function units in a processor, where a configuration phase involves a series of primary instruction words which comes from a translation of a program code being divided into a series of instruction word parts, and a program cycle subsequently involving instruction words which actuate the processor being generated in the full instruction word length as a VLIW (Very Long Instruction Word) and being buffer-stored in an instruction word memory (cache).

To this end, various solutions are known which deal with a respective advantageous variant for the synthesis of a VLIW (Very Long Instruction Word) from the instruction words which arise during the program cycle.

A common feature of these solutions is that the primary instruction words resulting from a translation of the program code are generated as a series of divided instruction word parts.

A current VLIW is thus constructed from a limited number of function instruction words (FIW), each of these FIWs actuating precisely one function unit (FU) in the processor.

German patent specification DE 198 59 389 C1 characterizes the prior art for methods of the type mentioned at the outset.

In the case of this solution, the primary instruction words which are present in the program are divided into individual program words, which are also advantageously referred to as TVLIW (Tagged Very Long Instruction

Word) containers.

They are called TVLIW containers because the individual program word is made up not only of an information part, which is represented primarily by an FIW (Function Instruction Word), but also the details regarding the write and read row numbers of an instruction word memory which is to be used. The latter details represent a tag for the FIW.

In addition, the program word also includes the details regarding how to handle the respective content of the instruction word memory characterized in this manner, and these are thus represented by an operating code (Opcode).

In the case of the aforementioned method, the data length of the program to be processed in the processor is advantageously reduced in order to keep down the hardware complexity and hence the costs for implementing the respective processor.

In addition, various solutions are known which deal with a respective advantageous variant for the synthesis of a VLIW (Very Long Instruction Word) from the FIWs which arise during the program cycle.

Hence, the printed document 102 03 541.5 for the German patent application outlines the continuing prior art.

In the case of this prior art, the division of the primary instruction words which is carried out in a configuration phase is expanded by a subsequent methodological automatic similarity analysis, the result of which is that the instruction word parts which have been selected with particular similarity features (periodic property) and hence can be used repeatedly are combined.

This series of instruction word parts is used in the subsequent processing phase to produce the VLIW with an operating code which is common in this regard and a flag (which is valid for all members of the series) for its periodic property which has the number of members added to it.

In this way, this specific compression operation involves the performance, in the configuration phase, of the selection and flagging of the instruction word parts which are provided for buffer-storage in the execution phase and hence save processor performance when the same instruction word parts are reused.

With the increasing complexity of the processors and the demands on the processing speed, it becomes clear that it is necessary to achieve a higher level of compression when coding the instruction word parts and decoding them in order to produce the VLIW (Very Long Instruction Word), since increasing the processing speed in another way, e.g. by increasing the operating clock frequencies, hits physical boundaries.

It is an object of the invention to achieve an increase in the processor performance during the execution phase by increasing the level of compression of the primary instruction words into their divided instruction word parts regardless of specific features (periodicity) of the FIWs.

The object based on the invention is achieved by virtue of a first step involving a primary instruction word being divided, in the configuration phase, into the series of a particular number of instruction word parts which are used for constructing a respective VLIW during the execution phase.

In this case, a respective first and second FIW (Function Instruction Word part) is preceded with an associated first or second operating code. This thus determines how the cache's memory location taken up by the respective FIW is handled in the execution phase.

In addition, the respective first or second operating code is followed by an associated first or second tag which represents the information regarding which first or second FU actuates the respective FIW.

The first or second operating code and its associated first or second tag are respectively combined with the respective first and second FIWs to form the first and second TVLIW containers.

In this context, all of these represent the TVLIW.

A second step involves the respective available TVLIW being converted into an HVLIW in the configuration phase. A general header is put in front in the HVLIW.

When converting the TVLIW into the HVLIW, the latter with the code-compressed general header structure it contains replaces all functions of the TVLIW.

In one variant, the inventive object is achieved by implementing a "Command Code" mode of operation of the HVLIW and its associated general header. This general header stores the information, in coded form, which indicates all combinations regarding which first and second FIW (instruction word part) is provided, after decoding, in the execution phase, for actuating a respective first and/or second FU (function unit) in the processor.

In addition, the general header stores which first and/or second FIW takes up memory locations in the

cache and whether or which operations are to be executed with the respective memory content in the execution phase in the cache when constructing the VLIW.

The aim of this solution is for the desired compression of the instructions to be implemented in the "Command Code" mode of operation of the HVLIW by combining a plurality of FIWs and an associated combination of the details regarding which of the FUs is to be actuated by which FIW, and also which FIW takes up particular memory locations in the cache when the VLIW is constructed and which operations are then executed with the memory content of said memory locations in relation to other memory locations in the cache.

This saves memory space and conserves processor performance.

One advantageous form of the variant of the inventive manner of achieving the object is achieved by virtue of the first part of the general header being provided with a header mode which contains information about the "Command Code" mode of operation of the HVLIW and of the general header.

This is followed by a second part which contains the respective most needed combination regarding which of the respective FUs is actuated by which first or second FIW.

This most needed combination is stored in the dictionary as a coded table value.

A third part is connected as CE information (Cache Extra information) and contains a pointer which refers to a provided location in the dictionary.

The last part of the general header which is provided

is the supplementary information.

The general header is followed directly by the first and second FIWs needed for constructing the VLIW.

This inventive solution lays emphasis on providing the "Command Mode" mode of operation with a structured general header which is very flexible and supports all types of "Command Code". This is also intended to remain valid for further development and updates and to safeguard its compression options.

A further variant of the inventive manner of achieving the object is for a "reference instruction" mode of operation of the general header to be implemented in which the FIWs provided for constructing the VLIW in the execution phase are buffer-stored generally in the cache.

In this context, the associated header mode bears a correspondingly decodeable tag for this "reference instruction" mode of operation. The "reference instruction" mode of operation is initiated by a specific HVLIW.

The latter contains an address statement which is used to refer to a reference instruction.

In addition, the subsequent HVLIW, which likewise bears the tag for the "reference instruction" mode of operation, contains a relative address for the address provided by the reference.

This has a mask appended to it which represents the FUs which are to be excluded from the actuation.

In the case of this beneficial variant of the inventive solution, the implementation of the specific "reference

instruction" mode of operation of HVLIWs avoids the long instructions for the processor kernel, which turn out to be long even in the "Command Code" header mode, for example, because they need to be able to be used for a large number of FUs (Function Units).

As a result, respective start and end phases of the instructions are also required for actuating the basic constituents of the individual FUs.

On account of a large number of identical FIWs which are produced for actuating the FUs in the instructions, e.g. in digital signal processors (DSPs), it is obvious from knowledge of the instructions for the processor kernel that the respective start and end phases of the instructions are redundant for the respective FUs.

This redundancy is avoided by the inventive solution by virtue of the HVLIW which initiates the "reference instruction" mode of operation being used to prescribe an address statement as a reference.

In the subsequent HVLIW of the "reference instruction" mode of operation, said HVLIW's general header is used to communicate only a relative address statement which can be used to decode the necessary FIW in the execution phase.

After the general header, this HVLIW likewise indicates the first and/or second FU (function unit), for which this particular instruction is not intended to be used, in coded form.

As a mask, which excludes the actuation of FUs, this statement can be made much shorter than a statement of all FIWs which are to be actuated.

Hence, for HVLIWs, the respective start and end phases

of the FIW intended for actuating the FUs provided need to be indicated only once in the general header in the "reference instruction" mode of operation.

This saves memory space.

Since this compression does not require the respective complete start and end phases of the instructions to be processed during VLIW construction, the processor performance in the execution phase is consequently likewise under less of a strain as well.

One specific variant of the inventive manner of achieving the object which implements the "reference instruction" general header's mode of operation in locally beneficial fashion is for the specific HVLIW which initiates the "reference instruction" mode of operation to refer, as a contained address statement, to the preceding HVLIW.

One further specific variant of the inventive manner of achieving the object which implements the "reference instruction" general header's mode of operation in globally beneficial fashion is for the specific HVLIW which initiates the "reference instruction" mode of operation to refer, as a contained address statement, to a general address.

One advantageous extension to the manner of achieving the inventive object specifically for the "Command Code" mode of operation of the HVLIW is for the execution phase to involve the HVLIW being decoded in a decoder1 which is equipped with a header decoder, a CMDT (Command Code Decompression Table), a cache and a cache miss repair logic unit, the HVLIW being available in buffer-stored form.

The header decoder identifies the "Command Code" mode

of operation of the general header from the header mode stored therein.

In addition, the identified header mode is taken as a basis for decompressing the values of the FU-C which are provided in the general header by means of a comparison with the CMDT and in conjunction with the CE information which is likewise taken from the general header.

The identified header mode is taken as a basis for processing the supplementary information in the general header.

Possible incorrect access during buffer-storage in the cache (cache miss) is remedied by the execution of an error handling routine in the cache miss repair logic unit.

Finally, the valid VLIW is provided at the output of the decoder1.

The invention will be explained in more detail with reference to an exemplary embodiment.

Figure 1 shows a block overview showing the compression steps which need to be executed in the configuration phase in order to convert the TVLIW 1 into the HVLIW 10 in the "Command Code" mode of operation, in line with the invention.

Figure 2 shows a block overview of the inventive decoder1 23 which, during the execution phase, decompresses the compressed HVLIW 10 into the VLIW 22 in the "Command Code" mode of operation and decodes it, in order to actuate the processor 21 in this manner.

As can be seen in Figure 1, in the configuration phase

the starting point for the inventive compression is the presence of the TVLIW 1. In the exemplary embodiment, this comprises the first and second TVLIW containers 11; 12.

The respective first or second TVLIW container 11; 12 is available with its constituents, the first or second operating code 2; 5, the first or second tag 3; 6 and the first or second FIW 4; 7.

In the order which arises, a respective TVLIW container is supplied to the code converter 18 and at the same time the code analyzer 8 ascertains the combination of the three constituents of a TVLIW container in terms of the frequency of their occurrence in relation to the further TVLIW containers of the respective TVLIW 1 through comparison with the details in the dictionary 9.

These details are made available to the code converter 18. The latter codes the general header 13 therefrom according to the mode of operation provided and logically combines it with the respective first or second FIW 4; 7, which are taken from the first and second TVLIW containers 11; 12 provided in succession.

When all the TVLIW containers of the TVLIW 1 have been processed, the structured general header 13 is provided and is available in the header mode 14, FU-C information 15, CE information 16 and supplementary information 17 parts. The general header 13 is put in front of the series of first and second FIWs 4; 7. A now complete HVLIW 10 is thus stored in the memory.

Subsequently, a further TVLIW 1 may be compressed.

The end of the inventive compression is reached when all of the TVLIWs 1 have been converted into a respective HVLIW 10.

As can be seen in Figure 2, in the execution phase the use of the inventive decoder1 (23) for decompressing/decoding the HVLIW 10 is triggered after the instructions have been buffer-stored (fetched) and hence upon provision of the HVLIW 10 and decoding of its header mode 14 using the available "Command Code" mode of operation.

Subsequently, the general header 13, as a constituent of the HVLIW 10, is buffer-stored in its constituents in the cache 26 and is decoded using the header decoder 24.

First, the first part of the general header 13, the header mode 14, is used to identify its mode of operation, and the decoder1 23 is set accordingly.

The second part of the general header 13, the FU-C information 15, provides the information for the first and second FUs 19; 20 regarding which of the first and second FIWs 4; 7 need to be taken into account by the CMDT 25.

From the third part of the general header 13, the CE information, the area of the CMDT 25 which is to be taken into account is processed. The supplementary information 17 is taken from the last part of the general header 13.

Any incorrect access to the cache 26 which is identified is remedied by the cache miss repair logic unit 27.

This information is used to construct the VLIW 22 by arranging the respective first and/or second FIWs 4; 7 in the VLIW 22 in the decoded order and position in which the first or second FU 19; 20 is subsequently

actuated on the processor 21 accordingly.

List of Reference Symbols

1 TVLIW (Tagged Very Long Instruction Word)
2 First operating code
3 First tag
4 First FIW (Function Instruction Word part)
5 Second operating code
6 Second tag
7 Second FIW
8 Code analyzer
9 Dictionary
10 HVLIW (Headed Very Long Instruction Word)
11 First TVLIW container
12 Second TVLIW container
13 General header
14 Header mode
15 FU-C information (Function Unit Combination information)
16 CE information (cache Extra information)
17 Supplementary information
18 Code converter
19 First FU (Function Unit)
20 Second FU (Function Unit)
21 Processor
22 VLIW (Very Long Instruction Word)
23 Decoder1
24 Header decoder
25 CMDT (Command Code Decompression Table)
26 Cache
27 Cache miss repair logic unit